

IN THE CLAIMS

Claims 34-37 have been amended. All pending claims are reproduced below.

1-17. (Canceled)

18. (Previously Presented) A method for compiling a functional description expressed in an interpretive, algorithmic language into target code for selected hardware, the method comprising the steps of:

parsing the functional description expressed in the interpretive, algorithmic language with at least one undeclared variable into an abstract syntax tree;

inferring a type and dimension for the undeclared variable by analyzing the usage of the undeclared variable in the abstract syntax tree;

assigning the inferred type and a dimension to the undeclared variable;

transforming compound statements in the abstract syntax tree into a series of single statements; and

translating the abstract syntax tree into a register transfer level format.

19. (Previously Presented) The method for compiling a functional description of claim 18, further comprising the steps of:

receiving a user directive file including at least one user defined directive selected from the group consisting of constraint directives, assertions, and compiler hints; and

annotating the functional description according to the user directive file.

1 20. (Previously Presented) The method for compiling a functional description
2 of claim 18, further comprising the steps of:

3 analyzing a value range of the at least one undeclared variable; and

4 assigning a required precision for the at least one undeclared variable.

1 21. (Previously Presented) The method for compiling a functional description
2 of claim 20, further comprising the step of:

3 parsing a real undeclared variable into an integer part and a fractional part,

4 wherein said real undeclared variable is one of said at least one undeclared variable.

1 22. (Previously Presented) The method for compiling a functional description
2 of claim 18, further comprising the steps of :

3 analyzing array access patterns across loop iterations; and

4 replacing a statement in a loop including a memory access with multiple

5 statements including the memory access to reduce the number of individual memory

6 accesses.

1 23. (Previously Presented) The method for compiling a functional description
2 of claim 18, further comprising the steps of:

3 analyzing compound loop structures to identify pipeline opportunities; and

4 applying the pipeline algorithm to pipeline opportunities to generate nodes

5 corresponding to the loop body, predicate nodes corresponding to loop conditional

6 statements, and a schedule for scheduling pipeline operations.

1 24. (Previously Presented) The method for compiling a functional description
2 of claim 18, wherein the step of transforming compound statements in the abstract syntax tree
3 into a series of single statements comprises the step of:

4 expanding a matrix operation into at least one loop.

1 25. (Previously Presented) The method for compiling a functional description
2 of claim 18, wherein the step of transforming compound statements in the abstract syntax tree
3 into a series of single statements comprises the step of:

4 deconstructing a compound statement into at least one simple statement.

1 26. (Previously Presented) A system for compiling a functional description
2 expressed in an interpretive, algorithmic language into target code for selected hardware
3 comprising:

4 a parser for parsing the functional description expressed in the interpretive,
5 algorithmic language with at least one undeclared variable into an abstract syntax tree;

6 a type-shape analyzer, coupled to the parser, for inferring a type and a dimension
7 to the undeclared variable by analyzing use of the undeclared variable in the abstract
8 syntax tree;

9 a statement deconstructor, coupled to the type-shape analyzer, for transforming a
10 compound statement in the abstract syntax tree into at least one simple statement; and

11 a translator, coupled to the statement deconstructor, for translating the abstract
12 syntax tree into a register transfer level format.

1 27. (Previously Presented) The system for compiling a functional description
2 of claim 26, further comprising:

3 a user directive file, coupled to the parser, for annotating the functional
4 description with at least one user defined directive selected from the group consisting of
5 constraint directives, assertions, and compiler hints.

1 28. (Previously Presented) The system for compiling a functional description
2 of claim 26, further comprising:

3 a precision analyzer, coupled to the type-shape analyzer, for determining the
4 precision of the at least one undeclared variable.

1 29. (Previously Presented) The system for compiling a functional description
2 of claim 28, further comprising:

3 a real number parser, coupled to the precision analyzer, for parsing a real number
4 into an integer part and a fractional part.

1 30. (Previously Presented) The system for compiling a functional description
2 of claim 26, further comprising:

3 a memory access optimizer, coupled to the statement deconstructor, for analyzing
4 array access patterns across loop iterations and replacing a statement in a loop including a
5 memory access with multiple statements including the memory access to reduce the
6 number of individual memory accesses.

1 31. (Previously Presented) The system for compiling a functional description
2 of claim 26, further comprising:

3 a pipeline optimizer, coupled to the statement deconstructor, for analyzing
4 compound loop structures to identify pipeline opportunities and applying the pipeline
5 algorithm to pipeline opportunities to generate nodes corresponding to the loop body,
6 predicate nodes corresponding to loop conditional statements, and a schedule for
7 scheduling pipeline operations.

1 32. (Previously Presented) The system for compiling a functional description
2 of claim 26, wherein the statement deconstructor for transforming a compound statement in the
3 abstract syntax tree into at least one simple statement comprises:

4 a scalarizer, coupled to the type-shape analyzer, for expanding a matrix operation
5 into at least one loop.

1 33. (Previously Presented) One or more computer readable storage devices
2 having computer readable code embodied on said computer readable storage device, said
3 computer readable code for programming one or more computers to perform a method for
4 compiling a functional description expressed in an interpretive, algorithmic language into target
5 code for selected hardware, the method comprising the steps of:

6 parsing the functional description expressed in the interpretive, algorithmic
7 language with at least one undeclared variable into an abstract syntax tree;

8 inferring a type and dimension for the undeclared variable by analyzing the usage
9 of the undeclared variable in the abstract syntax tree;

10 assigning the inferred type and a dimension to the undeclared variable;
11 transforming compound statements in the abstract syntax tree into a series of
12 single statements; and
13 translating the abstract syntax tree into a register transfer level format.

1 34. (Currently Amended) ~~One or more computer readable storage devices having~~
2 ~~computer readable code embodied on said computer readable storage device, said computer~~
3 ~~readable code for programming one or more computers to perform a method for compiling a~~
4 ~~functional description of claim 33~~[The method of claim 33,] further comprising the steps of:

5 receiving a user directive file including at least one user defined directive selected
6 from the group consisting of constraint directives, assertions, and compiler hints; and
7 annotating the functional description according to the user directive file.

1 35. (Currently Amended) ~~One or more computer readable storage devices having~~
2 ~~computer readable code embodied on said computer readable storage device, said computer~~
3 ~~readable code for programming one or more computers to perform a method for compiling a~~
4 ~~functional description of claim 33~~[The method of claim 33], further comprising the steps of:

5 analyzing a value range of the at least one undeclared variable; and
6 assigning a required precision for the at least one undeclared variable.

1 36. (Currently Amended) ~~One or more computer readable storage devices having~~
2 ~~computer readable code embodied on said computer readable storage device, said computer~~

3 ~~readable code for programming one or more computers to perform a method for compiling a~~
4 ~~functional description of claim 33~~[The method of claim 33], further comprising the steps of:

5 analyzing array access patterns across loop iterations; and

6 replacing a statement in a loop with a memory access with multiple statements
7 with the memory access to reduce the number of individual memory accesses.

1 37. (Currently Amended) ~~One or more computer readable storage devices having~~
2 ~~computer readable code embodied on said computer readable storage device, said computer~~
3 ~~readable code for programming one or more computers to perform a method for compiling a~~
4 ~~functional description of claim 33~~[The method of claim 33], further comprising the steps of:

5 analyzing compound loop structures to identify pipeline opportunities; and

6 applying the pipeline algorithm to pipeline opportunities to generate nodes
7 corresponding to the loop body, predicate nodes corresponding to loop conditional
8 statements, and a schedule for scheduling pipeline operations.